

Cryptography CS 555

Lecture 20



Identification Schemes

The Disavowal Protocol of the Undeniable Signature Scheme

- To prove that y is not a signature of x , one prove that y is a signature of x' and $x' = d_1^{1/e_1} g^{-e_2/e_1} \neq x$
 1. Bob randomly chooses $e_1, e_2 \in \mathbb{Z}_q^*$
 2. Bob computes $c_1 = y^{e_1} \beta^{e_2} \pmod p$ and sends to Alice
 3. Alice computes $d_1 = c_1^{a^{-1} \pmod q} \pmod p$ and sends to Bob
 4. Bob verifies that $d_1 \neq x^{e_1} g^{e_2} \pmod p$
 5. Bob randomly chooses $f_1, f_2 \in \mathbb{Z}_q^*$
 6. Bob computes $c_2 = y^{f_1} \beta^{f_2} \pmod p$ and sends to Alice
 7. Alice computes $d_2 = c_2^{a^{-1} \pmod q} \pmod p$ and sends to Bob
 8. Bob verifies that $d_2 \neq x^{f_1} g^{f_2} \pmod p$
 9. Bob verifies that $(d_1 g^{-e_2})^{f_1} \equiv (d_2 g^{-f_2})^{e_1} \pmod p$

Lecture Outline

- Identification schemes
 - passwords
 - one-time passwords
 - challenge-response
 - zero knowledge proof protocols



Authentication

- **Data source authentication (message authentication)**: a message is generated by a specific party.
- **Entity authentication (identification)**: the process whereby one party (the verifier) is assured of the identity of a second party (prover) involved in a protocol



Requirements of Identification Protocols

- Requirements of identification protocols
 - for honest prover A and verifier B, A is able to convince B
 - no other party can convince B
 - in particular, B cannot convince C that it is A
- Kinds of attackers
 - passive and replay
 - active, man in the middle
 - the verifier

Properties of Identification Protocols

- Computational efficiency
- Communication efficiency
- Security requirement of communication channels
- Trust in verifier
- Storage of secrets
- Involvement of a third party
- Nature of trust in the third party
- Nature of security: provable security

Authentication Using Fixed Passwords

- Prover authenticates to a verifier using a password.
- Require secure communication channels
- Total trust in verifier
- Passwords must be kept in encrypted form or just digests of passwords are kept.
- Attacks:
 - Replay of fixed passwords
 - Online exhaustive password search
 - Offline password-guessing and dictionary attacks

Unix crypt Algorithm

- Used to store Unix passwords
- Information stored in `/etc/passwd` is:
 - Iterated DES encryption of 0 (64 bits), using the password as key
 - 12 bit random salt taken from the system clock time at the password creation
- Unix use salting to change the expansion function in DES
 - to make dictionary attacks more difficult.
 - also to prevent use of off-the-shelf DES chips

One-time passwords

- Each password is used only once
 - Defend against passive adversaries who eavesdrop and later attempt to impersonate
- Variations
 - shared lists of one-time passwords
 - challenge-response table
 - sequentially updated one-time passwords
 - one-time password sequences based on a one-way function

Lamport's One-Time Password

Stronger authentication than password-based

- One-time setup:
 - A selects a value w , a hash function $H()$, and an integer t , computes $w_0 = H^t(w)$ and sends w_0 to B
 - B stores w_0
- Protocol: to identify to B for the i^{th} time, $1 \leq i \leq t$
 - A sends to B: $A, i, w_i = H^{t-i}(w)$
 - B checks $i = i_A, H(w_i) = w_{i-1}$
 - if both holds, $i_A = i_A + 1$

Challenge-Response Protocols

- Goal: one entity authenticates to other entity proving the knowledge of a secret, 'challenge'
- Time-variant parameters used to prevent replay, interleaving attacks, provide uniqueness and timeliness : nonce (used only once)
- Three types:
 - Random numbers
 - Sequences
 - Timestamp

Challenge-Response Protocols

- **Random numbers:**
 - pseudo-random numbers that are unpredictable to an adversary;
 - need strong pseudo-random strings;
 - must maintain state;
- **Sequences:**
 - serial number or counters;
 - long-term state information must be maintained by both parties+ synchronization
- **Timestamp:**
 - provides timeliness and detects forced delays;
 - requires synchronized clocks.

Challenge-response based on symmetric-key encryption

- Unilateral authentication, timestamp-based
 - A to B: $E_K(t_A, B)$
- Unilateral authentication, random-number-based
 - B to A: r_B
 - A to B: $E_K(r_B, B)$
- Mutual authentication, using random numbers
 - B to A: r_B
 - A to B: $E_K(r_A, r_B, B)$
 - B to A: $E_K(r_B, r_A)$

Challenge-Response Protocols Using Digital Signatures

- unilateral authentication with timestamp
A → B: cert_A, t_A, B, S_A(t_A, B)
- unilateral authentication with random numbers
A ← B: r_B
A → B: cert_A, r_A, B, S_A(r_A, r_B, B)
- mutual authentication with random numbers
A ← B: r_B
A → B: cert_A, r_A, B, S_A(r_A, r_B, B)
A ← B: cert_B, A, S_B(r_B, r_A, A)

Zero-Knowledge Protocols

- **Motivation:**
 - Password-based protocols: when Alice authenticates to a server, she gives her password, so the server can then impersonate her.
 - Challenge-response improves on this, but still reveals partial information.
- **Zero-knowledge protocols:** allows a prover to prove that it possesses a secret without revealing any information of use to the verifier.

Fiat-Shamir ID protocol (ZK Proof of knowledge of square root modulo n)

- System parameter: $n=pq,$
- Public identity: $v \quad v = s^2 \text{ mod } n$
- Private authenticator: s
- Protocol (repeat t times)
 1. A: picks random r in Z_n^* , sends $x=r^2 \text{ mod } n$ to B
 2. B checks $x \neq 0$ and sends random c in $\{0,1\}$ to A
 3. A sends y to B, where If $c=0$, $y=r$, else $y=rs \text{ mod } n$.
 4. B accept if $y^2 \equiv xv^c \text{ mod } n$

Observations on the Protocol

- Multiple rounds
- Each round consists of 3 steps
 - commit
 - challenge
 - respond
- If challenge can be predicted, then cheating is possible.
 - cannot convince a third party (even if the party is online)
- If respond to more than one challenge with one commit, then the secret is revealed.

Zero Knowledge Proofs

- A kind of interactive proof system
 - proof by interaction
- Involves a prover and a verifier
- Proving without revealing any other information

Two Kinds of Zero-Knowledge Proofs

- ZK proof of a statement
 - convincing the verifier that a statement is true without yielding any other information
 - example of a statement, a propositional formula is satisfiable
- ZK proof of knowledge of a secret
 - convincing the verifier that one knows a secret, e.g., one knows the square root modulo $N=pq$

Properties Zero-Knowledge Proofs

- Properties of ZK Proofs:
 - completeness
 - honest prover who knows the secret convinces the verifier with overwhelming probability
 - soundness
 - no one who doesn't know the secret can convince the verifier with nonnegligible probability
 - zero knowledge
 - the proof does not leak any additional information
- How to formalize soundness and ZK?

Formalizing the Soundness Property

- The protocol should be a “proof of knowledge”
- A knowledge extractor exists
 - that given a prover who can successfully convince the verifier, can extract the secret

Formalizing ZK property

- For every possible verifier algorithm, a simulator exists
 - taking what the verifier knows before the proof, can generate a communication transcript that is indistinguishable from one generated during ZK proofs
 - honest verifier ZK considers only the verifier algorithm in the protocol
- Three kinds of indistinguishability
 - perfect (information theoretic)
 - statistical
 - computational

Schnorr Id protocol (ZK Proof of Discrete Log)

- System parameter: p, q, g
 - $q \mid (p-1)$ and g is an order q element in Z_p^*
- Public identity: v
- Private authenticator: s $v = g^{-s} \bmod p$
- Protocol
 1. A: picks random r in $[1..q]$, sends $x = g^r \bmod p$,
 2. B: sends random challenge c in $[1..2^t]$
 3. A: sends $y = sc + r \bmod q$
 4. B: accepts if $x = (g^y v^c \bmod p)$

Security of Schnorr Id protocol

- probability of forgery: $1/2^t$
- soundness:
- ZK property
 - honest verifier ZK
 - not ZK if $2^t > \log n$ is used

Converting Interactive ZK to Non-interactive ZK

- The only interactive role played by the verifier is to generate random challenges
 - challenges not predictable by the prover
- The same thing can be done using one-way hash functions

Interactive ZK Implies Signatures

- Given a message M , replace the random challenge of the verifier by the one-way hash $c=h(x||M)$

Schnorr Signature

Key generation (as in DSA, $h:\{0,1\}^* \rightarrow \mathbb{Z}_q$)

- Select a prime q
- Select $1 \leq a \leq q-1$
- Compute $y = g^a \text{ mod } p$

Public key: (p, q, g, y)

Private key: a

Schnorr Signature

Signing message M

- Select random secret k , $1 \leq k \leq q-1$
- Compute
$$r = g^k \bmod p,$$
$$\mathbf{e = h(M || r)}$$
$$\mathbf{s = ae + k \bmod q}$$
- Signature is: (s, e)

Schnorr Signature

Verification

- Compute

$$v = g^s y^{-e} \text{ mod } p,$$

$$e' = h(m \parallel v)$$

- Valid iff $e' = e$

Summary

- Identification schemes
 - passwords
 - one-time passwords
 - challenge-response
 - Zero knowledge proof protocols
- Readings:
 - Handbook Chapter 10



Next ...

- Key establishment protocols
- Readings:
 - Handbook Chapter 12

